
DARPA Urban Challenge 2007

Team CarOLO

Technische Universität Carolo-Wilhelmina zu Braunschweig

C. Basarke¹, C. Berger¹, K. Cornelsen², M. Doering³, J. Effertz²,
K. Homeier³, C. Lipski⁴, T. Nothdurft⁵, J. Wille²

¹Institute of Software Systems Engineering,

²Institute of Control Engineering,

³Institute of Operating Systems and Computer Networks,

⁴Institute of Computer Graphics,

⁵Institute of Flight Guidance

of the

Technische Universität Carolo-Wilhelmina zu Braunschweig

38106 Braunschweig, Germany

team-carolo@ibr.cs.tu-bs.de

<http://carolo.tu-bs.de>

Submission date: June 1, 2007

“DISCLAIMER: The information contained in this paper does not represent the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA) or the Department of Defense. DARPA does not guarantee the accuracy or reliability of the information in this paper.”

Executive Summary

The interdisciplinary Team CarOLO of the Technical University of Braunschweig successfully developed an autonomous vehicle for urban environments based on a Volkswagen Passat. The architecture is derived from a strict linear signal flow, thereby minimizing module dependencies and decoupling the development process. Our novel approach to multi-sensor fusion of lidar, radar, and laser scanner data allows the tracking of multiple and diverse dynamic targets. These fusion objects are processed, weighted and stored in a highly optimized digital map. The artificial intelligence module "driving" Caroline is an enhancement of the proven DAMN architecture, which we completely redesigned to meet the requirements of urban environments. A sophisticated, multi-level testing strategy including functional and non-functional module tests assures software quality. Additional online tests of digital map and artificial intelligence have been enabled through the development of a simulation engine. Another novel aspect of our architecture is the deeply integrated safety concept. Integrity of vehicle, hardware, and software is constantly monitored. Any kind of malfunction or failure leads to an immediate total and safe stop.

1 Introduction

Team CarOLO was formed in June 2006 by some 35 students and research associates at the Technical University of Braunschweig. Its mission was to build an autonomous robot called "Caroline" to compete in the DARPA Urban Challenge 2007. The team includes members from several institutes and different faculties, each addressing different aspects of the project.

The Institute of Operating Systems and Computer Networks is responsible for the infrastructure in and outside "Caroline." It is primarily responsible for design and implementation of our artificial intelligence and planning algorithms. The Institute of Flight Guidance deals with the precise and highly available localization with respect to an earth-fixed reference frame. Additionally they are in charge of a digital map module, as well as collecting and predicting static and dynamic obstacles. The Institute of Control Engineering is responsible for Caroline's sensor system, data fusion, vehicle control, safety concept, and low level actuators. The camera based vision module provided by the Institute of Computer Graphics. The module extracts lane markings and drivable areas supported by laser scanner data. The Institute of Software Systems Engineering is in charge of project organization, fundamental software framework, and multi-level test processes. In addition to other sponsors and technical partners, the team is assisted by the IAV GmbH.

Approaching the requirements associated with the Urban Challenge, we started analyzing the problems and structuring the tasks established by DARPA guidelines. In order to take advantage of the individual strengths of our heterogeneous team, we decided to create an application structure based on linear signal flow between the modules as shown in Figure 1. This approach enabled us to highly parallelize the development process and to facilitate interdisciplinary teamwork based on predefined interface structures.

Several support processes, e.g. software framework, simulator, hardware environment, and testing facilities were launched early to reduce development time. This allowed us to implement, test, and optimize individual software modules even prior to their integration into the overall system.

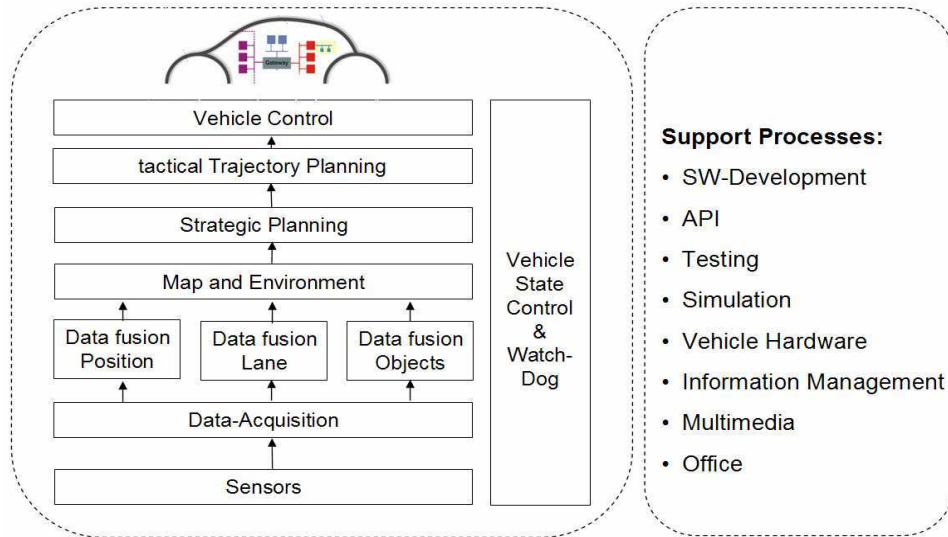


Figure 1 Team CarOLO's approach

In the following we describe our approach in detail. Starting with the requirement analysis in which the overall design is derived from the problem to be solved, we continue with a brief overview of the architecture followed by detailed descriptions of our functional modules. In the next section preliminary results are discussed and the system's performance is evaluated where applicable. Moreover, we present our testing process that verifies and ensures the safety and reliability of our design.

2 Requirements Analysis

2.1 Problem to be Solved

DARPA guidelines require the autonomous robot to comply with four evolutionary steps of increasing complexity:

- **Basic Navigation:** Besides other basic behaviors, the vehicle shall be capable of proceeding to identified checkpoints in a sequentially correct order within a defined route network, adhering to speed limits, avoiding static and dynamic obstacles, staying in lane, detecting stop lines and performing u-turns.
- **Basic Traffic:** The autonomous robot must comply with basic traffic rules, including maintaining safe following distances, respecting the right-of way at intersections and realizing correct queuing behavior.
- **Advanced Navigation:** More complex scenarios, such as obstacle fields, parking lots, dynamic route-replanning, road following and GPS-outages have to be tackled.
- **Advanced Traffic Interactions** of high complexity, including merging into moving traffic, left turns onto frequented roads and traffic jams have to be managed.

2.2 Technical Requirements

In the context of the requirements established by the Urban Challenge competition, a number of technical requirements can be identified:

- **Robotic Perception:** Overall, a 360 degree field of view must be covered by Caroline's sensors in order to master all possible situations. Certain areas, however remain that demand special treatment: The front section of the vehicle must be covered over a long range in order to enable appropriate maneuvers while following and passing in moving traffic. In order to deal with intersections, the field of view must be expanded to the right and left to account for cross traffic. The rear of the car must be covered for safe lane-changes and situations in which backing up in traffic is necessary. In addition to these requirements, the sensor system must be capable of precisely recognizing available parking spaces. For each requirement in the DARPA guidelines, we created a catalogue of possible scenarios to derive the necessary detection ranges and opening angles of our sensor systems, e.g. two vehicles being on a direct collision course with 30 mph, overtake-maneuvers at maximum permissible speed or similar configurations.

- **Robotic Control and Localization:** Caroline must be capable of following GPS-waypoints provided by the RNDF / MDF. If GPS outages occur, a backup solution must come into play. The control system must provide for safe vehicle travel within all situations, while taking vehicle dynamics and actuator limitations into account. All actuating variables normally managed by the human driver have to be controlled by the robotic systems with at least the same dynamics and limits, i.e. throttle, brake, steering and transmission selector lever position. Both driving directions have to be covered. To enable precise lateral and longitudinal control, the vehicle's position must be known with a precision down to several centimeters.

- **Artificial Intelligence:** While the main focus of all other modules is data processing and data distribution, the AI module has to interpret complex situations given the abundance of available sensor information. The complexity common in an urban environment has to be mastered. Different vehicle behaviors including tracking, obstacle avoidance, interaction with traffic and route identification have to be appropriately accomplished while obeying all applicable traffic rules.

- **Safety System:** If necessary, the autonomous robot must be stopped and powered down to a safe inactive state as quickly as possible to prevent personal injury and property damage. An E-Stop system providing wireless shutdown and resume has to be installed. Several emergency shutdown access points must be installed inside and outside the vehicle, visual as well as acoustic alarms to signal the autonomous driving mode are also required by DARPA.

2.3 Design Choices and Decisions

In our approach, the principal issues that drive design choices are safety, reliability, performance and cost. Based on the previous requirements analysis, a multitude of decisions were necessary. In this section we present some selected examples and the analyses that lead to our decisions.

2.3.1 Computing System

An important choice in the design of Caroline was the selection of a suitable computing system. Two alternatives were evaluated: First, a monolithic approach with a central high performance server running all software applications. Second, a decentralized approach to run each application on its 'own' computer. The advantages of the monolithic approach are a low overall power consumption of roughly 300W, less weight, simpler maintenance, and a reduction of overall system complexity. On the other hand, redundancy improves fault tolerance. Moreover, off-the-shelf high performance servers are designed to operate in air-conditioned server rooms. In an automotive environment, however, the computing system must cope with high temperatures, shock and vibration, as well as electromagnetic interference. Since reliability is imperative, we decided on the second approach, despite the fact that overall power consumption increases to 460W and is thus significantly higher than the 300W of the monolithic approach. A rack in Caroline's trunk carries seven automotive computers and two Gigabit Ethernet switches. These components were carefully evaluated for reliable operation under automotive conditions and power-efficiency. The six computers are Pentium M based, each consuming less than 50W under full load. The remaining system is equipped with a Core 2 Duo CPU and a high performance GPU, since it runs the vision system that requires greater computing power. Its power consumption is 160W at maximum.

2.3.2 Sensor Concept

Given the requirements from section 2.2, we subdivided Caroline's sensor concept into two main categories: Object based processing and raw-data based processing. This design choice was influenced mainly by the prevailing market situation concerning the availability sensor technology at reasonable prices and as well by the very short time frame in which our development work had to be carried out.

The first sensor category is based on standard hardware from automotive applications, including adaptive cruise control and similar applications. These devices are thus equipped with built-in internal object tracking stages that assign fixed object numbers to detected features during the surveillance period. We use these sensors to detect approaching and receding traffic as well as stationary obstacles in or next to the driving corridor. One benefit of the internal tracking stages is that no raw-data processing is necessary, thereby significantly simplifying the development process while saving computing power. A centralized tracking and sensor fusion algorithm had to be created, however, to combine redundant sensor information into a single image of the vehicle's surroundings. As a consequence of the requirement analysis, a 360 degree field of view must be covered with emphasis on specific regions of higher interest and importance. Accordingly, we chose to create a nearly symmetric sensor concept in the front and rear of the vehicle. In earlier projects we were able to gain experience with different sensor technologies and develop an appreciation of their strengths and weaknesses. This experience led us to a

combination of three different measurement principles to gain additional redundancy and cancel-out sensor-specific clutter: Radar, Lidar and Laser Scanners.

The Table below summarizes the different sensor technologies and detection areas:

Sensor Type	Function	Range	Opening angle
Front Medium Range Radar	front traffic detection	150m	40 degrees
Rear Medium Range Radar	rear traffic detection	150m	40 degrees
Left Short Range Radar	left blind spot coverage	14m	140 degrees
Right Short Range Radar	right blind spot coverage	14m	140 degrees
Front Lidar	front traffic and obstacle detection	200m	12 degrees
Rear Lidar	rear traffic and obstacle detection	200m	12 degrees
Front Laser Scanners	front obstacle and traffic detection	80m	250 degrees
Rear Laser Scanner	rear obstacle and traffic detection	80m	180 degrees

The second sensor category is based on camera technology and laser scanners from an industrial application. The raw data derived from combined measurement will be used to detect the lane markings and the drivable area in front of the vehicle in order to handle obstacles that cannot be detected by the object based sensor system (e.g. potholes). As a further source of measurement data, the object-based laser scanners from the first category can be operated in a hybrid-mode, simultaneously delivering condensed track data and raw scan data using different interfaces.

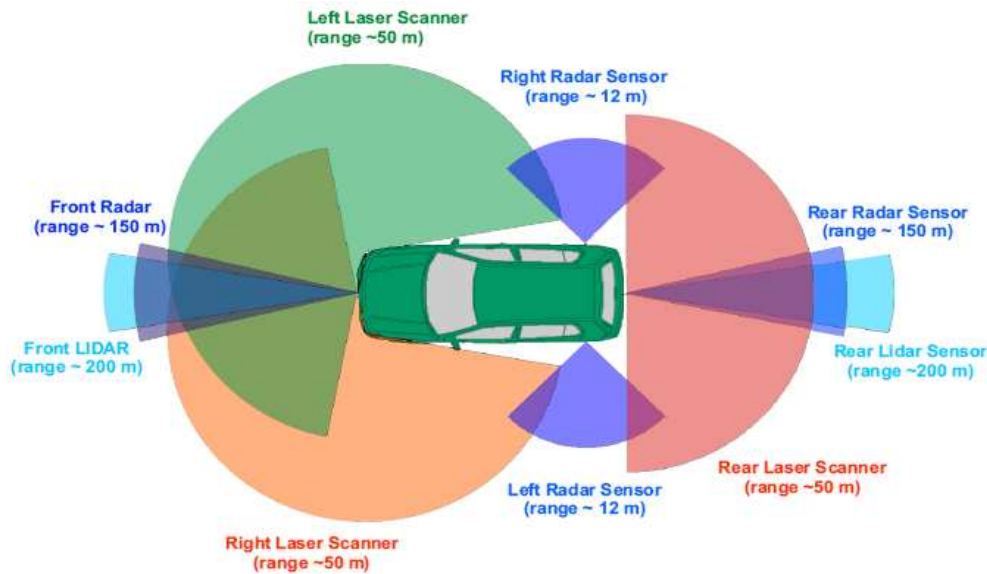


Figure 2 Sensor View Areas (not scaled)

2.3.3 Sensor Data Fusion

Caroline's data fusion architecture had to be closely connected to its data acquisition and sensor system design. Accordingly, we also split up the data fusion into two separate units covering the front and the back of the vehicle. We explored two alternative data fusion architectures in order

to deal with the high number of sensor objects being pushed into the data fusion stage by our sensor network:

In a parallel approach (Figure 3a), all sensors would have to be synchronized to a common moment of measurement and their readings would then be fused by a multi-sensor data-fusion algorithm such as Fisher's information filter or a multi-sensor Kalman filter [1], [3].

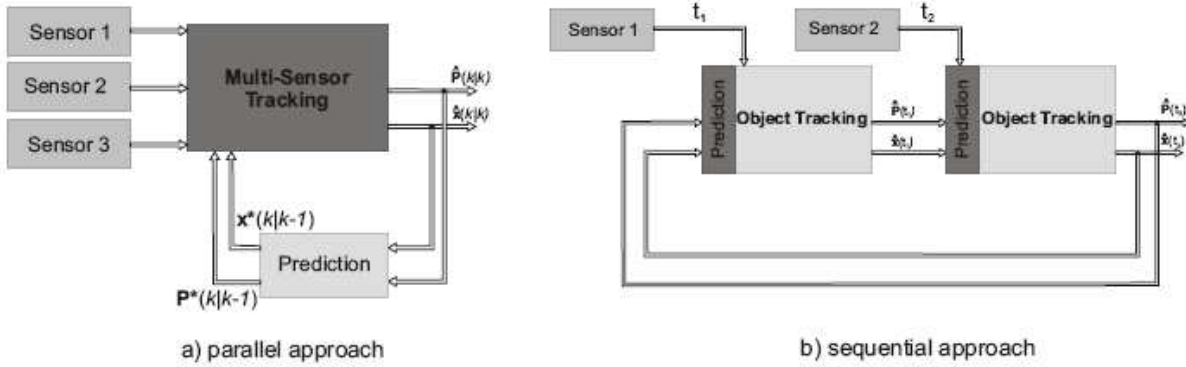


Figure 3 parallel vs. sequential data fusion

This, of course, implies that the sensors can be synchronized - which is not the case given standard automotive environmental devices that additionally come from different manufacturers. While it is possible to predict the sensor readings to a common fusion post-reception timeslot using simple movement models, this would add additional uncertainties and also create computational overhead.

In a sequential approach (Figure 3b), each sensor is fused "on time", i.e. the readings are pushed into the data fusion stage immediately after their reception and processed using a first-in first-out principle. There is no need to synchronize the sensors under this approach. Some extra computation time, however, will be required when retrieving the update stage of the state estimators more often within the tracking algorithm. Due to the splitting of data fusion into front and back section and economies of computing power today, we have chosen to implement the sequential approach that has one additional major advantage: Within the sequential hierarchy, additional sensors can be added easily without major changes in the data fusion architecture.

2.3.4 Digital Map

There are various mapping algorithms for real-time applications: probabilistic, incremental, object based and hybrid forms. The key issue in those different map implementations is the amount of data that has to be processed. Object-based map approaches significantly reduce data complexity for faster updates and lower latencies. We have chosen to use an object based map design, since our sensor and fusion concept is also mainly object-based. In further developmental steps, this approach will be extended to a hybrid probabilistic and object based implementation. Instead of storing all data received about Caroline's surroundings, we decided to store only key information in the road graph such as travel time between two waypoints or dead ends. We will

enrich the graph with more waypoints and lane information during missions. A detailed description is provided in section 4.4.

2.3.5 Artificial Intelligence

We first evaluated a high-level 'artificial intelligence' state-based approach similar to [9]. Early tests showed that modeling every state, the car could encounter and describing the correct state transitions resulted in unacceptable complexity. The system would fail if something unexpected occurred. In order to avoid this complexity and to maintain flexibility to handle unexpected situations, we enhanced the stateless DAMN-Architecture described by [10], making its decisions upon defined behaviors. On the one hand, it is not easy to precisely predict system reaction to different given inputs, but on the other hand, modularity and robustness will overcome this disadvantage, since the robot will always react in the way the defined behaviors are satisfied best.

2.3.6 Vehicle Platform and Low Level Actuators

A model 2006 Volkswagen Passat station wagon has been chosen for the vehicle platform, because this car offers the technical interfaces to control steering, braking, and throttle via the standard power train CAN-Bus. Therefore no further actuators to control the vehicle have been installed.

3 Architecture Overview

Caroline is a standard, model 2006 Volkswagen Passat, built to European specifications and equipped with a variety of sensors, actuators and computation units in order to serve as autonomous mobile robot. In front, two multi-level laser scanners, one lidar sensor and one radar sensor cover a field of view up to 200m for approaching traffic or stationary obstacles. In addition, three cameras detect and track road lane boundaries in order to allow precise lane keeping. Very similar to the vehicles front, one multi-level laser scanner, one medium range radar, one lidar and two radar-based blind-spot-detectors enable Caroline to detect obstacles at the rear.

The development process of Caroline is divided among a variety of institutes and specialization areas, combining faculties for computer science, mechanical and electrical engineering. Mirroring this internal structure, Caroline's architecture is grouped into eight main modules, interconnected with predefined interfaces as shown in Figure 5.

- Sensor Data Acquisition,
- Sensor Data Fusion,
- Image Processing,
- Digital Map,
- Artificial Intelligence,
- Vehicle Path Planning and Low Level Control,
- Supervisory Watchdog and Online-Diagnosis,
- Telemetry and Data Storage for Offline Analysis.

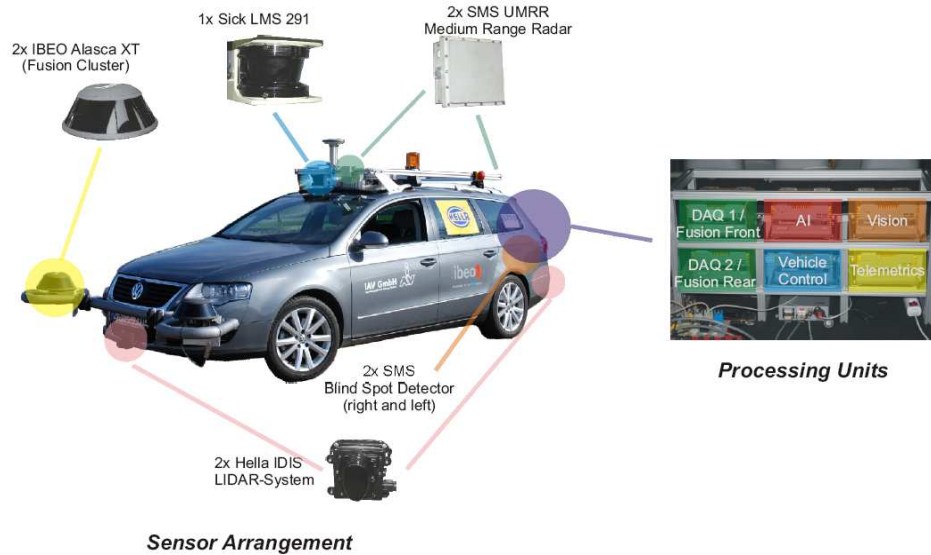


Figure 4 Vehicle Overview

An array of automotive PCs serves as the hardware platform for this distributed software architecture, with all internal communication based upon 1 GBit Ethernet. The access to Caroline's by-wire steering, brake and throttle system as well as to other low level actuators is provided through a CANLOG III command interface, which also connects to the vehicle's E-stop system to provide emergency stop functionality even if the complete software tree described as above should fail.

Not counting those lower level components described above, all computing and control hardware is based on industrial PC technology, thereby reducing hardware variety, simplifying error management as well as component replacement. Due to the linear signal flow between each function module intentionally chosen and the associated limitation of closed loops to a possible minimum, our team is able to develop different modules independently and with minimum interference.

Starting at the bottom of this linear flow, the data acquisition unit provides necessary hard- and software modules to collect and process incoming data from the radar, lidar and laser scanner sensors used for object recognition. Since all of the sensors used are standard components originating from contemporary automotive driver assistance systems, they are equipped with a Controller Area Network (CAN) communication interface. Taking into account the limitation of this bus standard regarding data throughput and determinism, a private sensor CAN has been chosen for each sensor to keep latencies small and avoid bus conflicts.

Incoming video data is sampled from the assigned IEEE 1394 interface, preprocessed and interpreted directly on the image acquisition PC in order not to heavily load the vehicles internal network with image data. The acquisition of GPS and INS data (vehicle ego state) has been moved directly into the real-time vehicle control unit in order to avoid even larger latencies within the closed loop dynamic control.

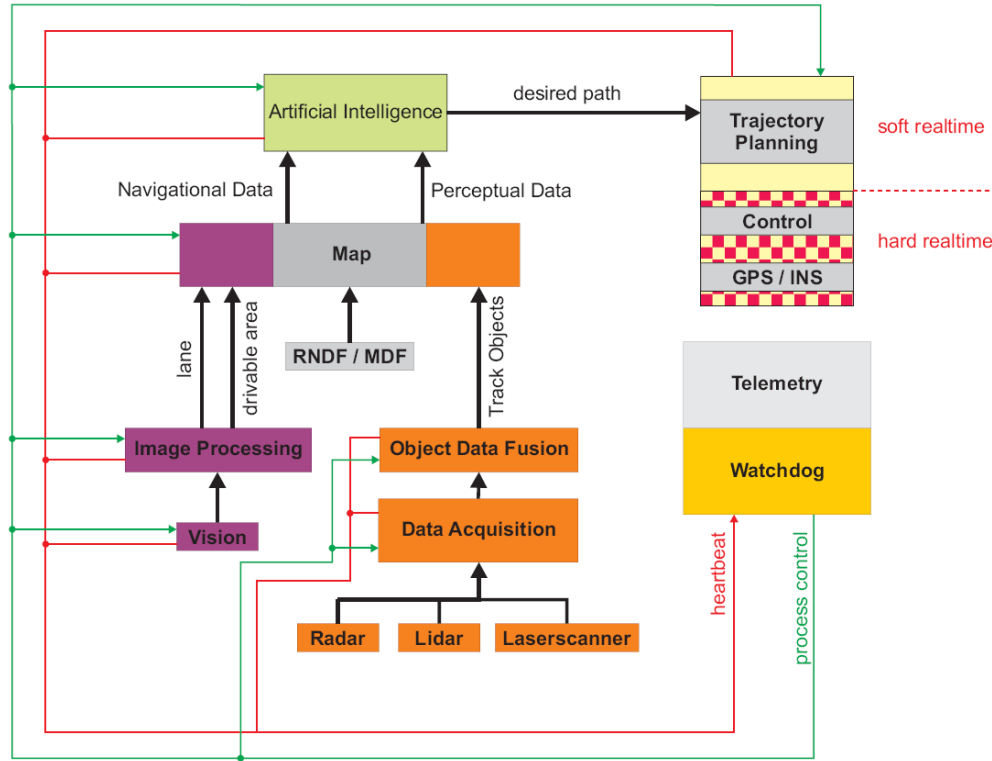


Figure 5 Software Architecture

Furthermore, following Caroline's signal flow, sensor data of all object-recognizing sensors is processed within a central sensor data fusion unit (see 4.2) which transmits the object-based artificial surroundings of the vehicle containing all (recognized) static and dynamic targets in Caroline's field of view to the map unit.

The digital map combines online environmental information with available offline information generated from MDF and RNDF provided during the mission. This combined data is the basis for our artificial intelligence module to generate driving decisions based on a Distributed Architecture for Mobile Navigation - scheme (DAMN) as proposed by [10].

The driving commands obtained, e.g. "follow a given road" are issued to the soft real time control module, which carries out trajectory generation and optimization based on driving dynamics of the vehicle. The driving trajectories generated are then passed along into hard real time control that addresses the vehicle actuators.

All modules previously described are supervised by a central watchdog process with the possibility to kill and restart one or several processes, computers or sensors independently. Thus, a maximum of self-healing capability is installed in Caroline's systems.

4 Final Realization

4.1 Sensor Data Acquisition

In accordance with Figure 5 and Figure 3, a central data acquisition layer has been created which carries out bus communication, data format conversion, time-stamping and data pre-processing. To equally load the computing hardware used, the rear and front systems have been split onto separate units. All measurement data is transformed into a common, earth-fixed coordinate system using the vehicle's own position and orientation from the GPS / INS measurement unit. All sensors have been connected to their central acquisition unit with the Controller Area Network (CAN). Due to the regulations regarding bandwidth and determined by the bus standard, a private sensor CAN has been chosen for each sensor to avoid further complications. The communication to each sensor is monitored so that malfunctions of the sensors' internal control units can be detected and rectified by powering off and on single sensor systems through Caroline's watchdog system via a specialized power supply interface. Three different data structures are used within Caroline's object detecting sensors, which will be treated differently within the data fusion unit:

- The radar sensors deliver point-shaped targets carrying an assigned velocity vector. Since the sensors are more suited to detect moving reflectors of significant size, such as, cars, trucks etc., they will not detect stationary targets with low reflectivity to the radio waves emitted.
- The lidar sensors internally consist of 16 independent infrared beams operating sequentially while measuring distances in each beam through the time-of-flight principle. The sensor's internal tracking is capable of extending objects into neighboring beams and therefore delivers objects in a line-shaped fashion described by left and right termination point forming a line perpendicular to the viewing axis.
- The laser scanners, the backbone of Caroline's sensor architecture, consist of four scan planes each, rotating at a frequency of 12.5 Hz. Compared to the rather simple object description provided by radar or lidar, they deliver a far more complex object model consisting of several contour points interconnected by lines and a common velocity vector for each target.

4.2 Sensor Data Fusion

Within Caroline's data fusion unit, incoming data from all perceptual object based sensors is associated and fused into a single artificial representation of the vehicle's surrounding. More detailed, four stages are performed to realize central object tracking and data fusion:

- Data Association
- Pre-tracking and Track Initialization
- Object Tracking and Data Fusion
- Track Management

4.2.1 Data Association

During Data Association, incoming sensor data is mapped onto existing tracks within Caroline's central track database. In a classical approach regarding point-shaped objects, it is possible to define a cost function

$$c(i, j) = \text{dist}(\text{Track}(i), \text{Measurement}(j)) \quad (1)$$

that counts for the similarity between track i and measurement j . Common approaches are the Euclidian point-distance or point-distance combined with velocity-dissimilarity. A gate is being defined around each track to cancel out measurements which cannot be originating from the track object regarded. If this gating does not cancel out all ambiguities, an optimization process (e.g. Nearest Neighbor or Hungarian Algorithm) is carried out to find the best match for each track.

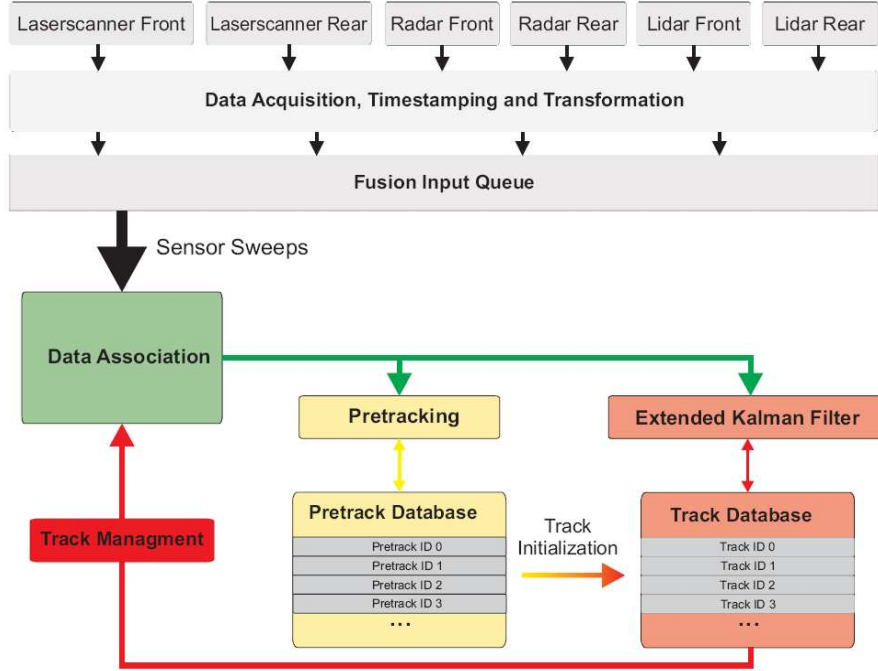


Figure 6 Fusion State Flow

4.2.2 Pretracking and Track Initialization

All readings that cannot be mapped within the Data Association stage are candidates for new tracks and will subsequently be passed into the track initialization stage. Within this stage, potential tracks are created temporarily and simple data association is carried out during each measurement cycle. If a potential track has been confirmed sequentially during a couple of measurement cycles, it is passed into the internal track database.

4.2.3 Object Tracking and Data Fusion

Caroline's data fusion is based on Extended Kalman Filters. Each track is described by an associated track state vector x :

$$\vec{x} = [x, y, v, a, \alpha, \omega]^T \quad (2)$$

with x , y , v , a , α , ω being x-position, y-position, velocity, acceleration, course angle and course angle velocity, respectively. Depending on the individual sensor type (radar, lidar or laser scanner), the measurement vector y consists of

$$\vec{y} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \quad \text{or} \quad \vec{y} = \begin{bmatrix} x \\ y \end{bmatrix} \quad (3)$$

with x , y , v_x , v_y being the measurements x-position, y-position, velocity x-component and velocity y-component, depending on whether the sensor is capable of delivering appropriate target velocity information or not. The track update process is then carried out by

$$\vec{x}^*(v+1|v) = \mathbf{A} \cdot \hat{\vec{x}}(v|v) \quad (4)$$

$$\mathbf{P}^*(v+1|v) = \mathbf{A}^T \cdot \hat{\mathbf{P}}(v|v) \cdot \mathbf{A} \quad (5)$$

$$\mathbf{S}(v+1) = \mathbf{C} \cdot \mathbf{P}^*(v+1|v) \cdot \mathbf{C}^T + \mathbf{R}(v+1) \quad (6)$$

$$\mathbf{K}(v+1) = \mathbf{P}^*(v+1|v) \cdot \mathbf{C}^T \cdot \mathbf{S}^{-1}(v+1) \quad (7)$$

$$\hat{\vec{x}}(v+1|v+1) = \vec{x}^*(v+1|v) + \mathbf{K}(v+1) \cdot [\vec{y}(v+1) - \mathbf{C} \cdot \vec{x}^*(v+1|v)] \quad (8)$$

$$\hat{\mathbf{P}}(v+1|v+1) = [\mathbf{I} - \mathbf{K}(v+1) \cdot \mathbf{C}] \cdot \mathbf{P}^*(v+1|v), \quad (9)$$

with matrices \mathbf{Q} , \mathbf{R} and \mathbf{S} being the system noise, measurement noise and innovation covariance, respectively. Matrices \mathbf{A} and \mathbf{C} are Jacobi-Matrices of the nonlinear time-transfer and measurement output functions for the system state \mathbf{x} .

4.2.4 Track management

The Track Management stage handles "dead" tracks that cannot be associated with any sensor reading when leaving Caroline's field of view. Since all tracks carry an update timestamp, the central track database can easily be queried for tracks with an exceeded update period and those tracks can subsequently be deleted.

4.2.5 Our extensions

While the classical tracking approach described works well for point-shaped objects common in aerospace applications (e.g. radar tracking), it does not handle objects with an extended object shape, which - even worse - is not detected constantly over time as Caroline passes by extended objects, e.g. walls. Therefore, we extended the Data Association and Object Tracking stages in order to take into account objects consisting of numerous contour points. The simple cost functions from section 4.2.1 had to be extended to incorporate point-to line, point-to-contour, line-to contour and contour-to-contour matching, representing the different internal sensor data structures. Additionally, more than one object from the same sensor can now be used to update a specific track, since it is quite possible that an extensive contour received by laser scanners can be seen as several objects by the radar or lidar.

The EKF update cycle has been modified, separating the object's state vector into contour point coordinates on the one hand and velocity, acceleration, course angle and course angle velocity on the other. Furthermore, an additional stage to handle track splitting and track merging has been included for more complex situations (e.g. passenger exiting a car).

4.2.6 Future work

Ongoing work is being carried out to improve the association process, carefully tune all gating threshold levels and tune the noise parameters within the Kalman filters for filter convergence in a minimum of time. Additional work has to be carried out to further suppress ghost objects by efficiently using the sensor redundancy in parts of the field of view. Furthermore, the suppression of artifacts induced through smaller depressions or knolls in the field of view, especially in the laser scanners' sensor, data must be carried out before the final event.

4.3 Image Processing

In order to stay in the travel lane, Caroline must gather and interpret data concerning the road markings. Using monochrome images, lanes and stop lines can be detected and tracked over time. Three cameras mounted on the front record images of the surrounding environment at 30 frames per second. These images are synchronized in terms of trigger time and shutter value. Knowing the intrinsic parameters, lens distortion effects can be removed from the images. For each camera, a projection matrix has been computed, which transforms the original images into the vehicle's coordinate system, resulting in an aerial view of the surrounding area (Figure 7). Since these transformed views are in the same coordinate system, they can easily be merged into a single panoramic camera image.

The advantages are as follows:

- the aperture angle is almost three times larger compared to the original images,
- angles and distances can be measured in the image, as it lies in the car reference frame,
- knowing the current position and orientation of Caroline, the position of given objects

in the global coordinate system can be easily transformed into the image and vice versa. Upon initialization, the system makes a naive initial estimate as to where road markings may be. This estimate is validated using scanlines. If they are placed on a road marking in a more or less perpendicular direction, they can detect this marking by searching for two distinctive features:

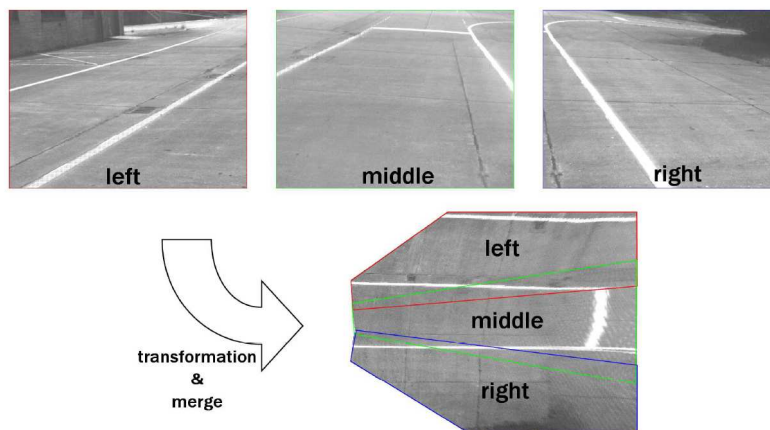


Figure 7: Images are transformed to an aerial view and then merged

- Road markings usually have a brighter color than the road itself. The median luminescence value of all pixels on the scanline is computed. This value is then subtracted from the luminescence values on the scanline. Now, all luminescence values significantly higher than 0.0 can be interpreted as a potential road marking. A special luminescence kernel is used to search for scanline regions containing a distinctive quantity of bright pixels.
- Along the edges of road markings, steep luminescence gradients can often be found. A one-dimensional Sobel kernel detects these gradients on the scanline. Two kernels are then applied to search for nearby gradients, the first kernel detects gradients left of the examined pixel, the second detects gradients on the right side.

For each pixel, the value of the luminance kernel and the two gradient kernels are multiplied. A high result indicates the presence of a road marking on the scanline. Several scanlines are used to detect a single yellow or white line on the road. As these lines are modeled as B-Splines, each scanline's task is to detect a contour point of the B-Spline. After initialization, the position in the global coordinate system of the contour points is tracked using a Kalman Filter. New contour points are added to the spline when the vehicle moves forwards or backwards. Stop lines are found the same way, the only difference is that a stop line is not modeled as a spline but as a straight line.

4.4 Digital Map

The map's tasks include the storage and management of an artificial view of the surrounding of our autonomous robot. For this purpose it receives objects from sensor data fusion and drivable terrain from the vision module. The acquired data is queried by the AI. The first approach consisted of a local rolling occupancy grid with time-decreasing cell values describing the probability of occupancy. Considering the range of Caroline's vision, the edge length of the squared grid is set to 400m. The whole grid is divided into subgrids and these again into cells with an edge length of 20cm. The subgrids allow rapid grid movement and efficient memory management. If the vehicle is moving out of a defined area in the middle of the grid, the grid rolls about one subgrid in the vehicle's direction. This procedure requires no memory allocation, because the subgrid's position is defined by an array of pointers. Therefore, it is only necessary to clear the memory of the moving subgrids and relocate them via the pointer array. Obstacles are placed into the grid by increasing cell values inside incoming fusion object contours. The obstacles size is increased, so that the vehicle can be regarded as point-shaped, speeding up collision detection. Being based on the theory of occupancy grids, which basically describes static environments, this approach cannot be used to handle moving targets. A different approach was to use fusion object data directly. Upon AI requests, fusion object positions are predicted up to the time of interest to check for possible collisions. A significant advantage of this approach is the considerably smaller quantity of data that must be managed during AI requests. Section 5 compares these approaches. The next step for the map module will be to process the information about potholes, small obstacles, and the drivable terrain detected by the vision module, which leads to a hybrid grid and object-based approach.

4.5 Artificial Intelligence

4.5.1 The DAMN-Architecture

In our approach, Caroline's movements are limited to two degrees of freedom: Lateral and longitudinal movements. Turning the steering wheel results in different circle-radii on which the car will move. Instead of the radii, the approach is based on the inverse curvature. A curvature of 0 represents driving straight ahead, while negative curvatures result in left and positive curvatures in right turns (see Figure 8). This curvature, as the most important thing to influence, is selected in an arbiter as described in the DAMN-architecture [10]. This architecture models each input as behavior, which gives a vote for each possible curvature. More behaviors can be added easily to the system, which makes it very modular and extendable. The following behaviors are considered:

- Follow waypoints: Simply move the vehicle from point to point as found in the RNDF.
- Stay in lane: Vote for a curvature that keeps the robot in the detected lane.
- Avoid obstacles: Vote for curvatures that keep Caroline as far away from obstacles as possible and prohibit curvatures leading directly into them.
- Stay on safe ground: Stay in drivable area detected by cameras and avoid potholes and small obstacles detected by laser scanners (planned).

All collected votes are weighted to produce an overall vote. The weights again are not fixed, they depend on factors such as distance to an intersection, presence of lanes, etc... Another arbiter controls the speed, influenced by different behaviors, which each provide a maximum speed. The arbiter simply selects the minimum of these speeds. Based on curvatures iteratively determined, we designed a drivable corridor for further processing by the next module in the chain, the path planner.

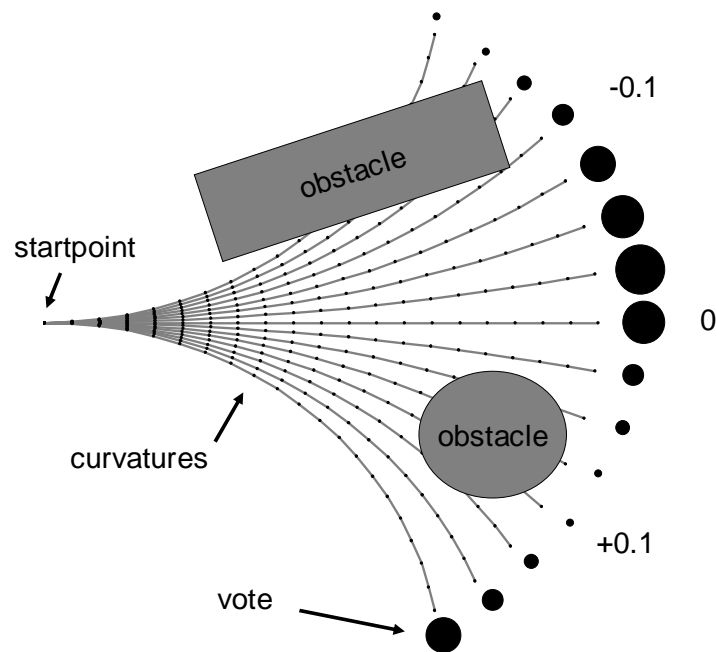


Figure 8: Curvature field: larger black circles represent better votes

4.5.2 Our Extensions to the DAMN-Architecture

Because we have to deal with more complex situations than the DAMN-architecture is designed to accommodate, our software detects situations where a special treatment is necessary. For example if we reach a stop line or a position in which a u-turn is required, we set an interrupt point, at which the arbiter transfers control to a specialized mechanism. At an intersection, for example, the vehicle comes to a complete stop at the stop line as recognized by the cameras and waits until the intersection observer votes to go. This observer initially finds intersections by parsing the RNDF and analyzing exit and entry points. Intersections are continuously modified as they come into camera focus. When reaching the stop line, the intersection observer decides based on the fused sensor data if there are other vehicles that have the right-of-way. If so, Caroline lets them pass and for the intersection to clear. The control is then given back to the arbiters to continue in normal driving mode.

4.5.3 Future Work

It is not an easy task to determine the optimum weighting for a specific situation for different votes. As the system grows this problem will become more complex. We will design a training-mode in which a human driver steers the car and collect all data that may influence AI decisions. Finally we will preprocess the data to find a combination of votes, which best fits human driving practices.

4.6 Vehicle Path Planning and Low Level Control

As previously mentioned, Caroline can be fully controlled via the CAN-bus using by-wire steering, gas and brakes. A vehicle path must first be planned. Therefore, the trajectory planner was introduced into the system to determine a dynamically optimized trajectory based on a drivable corridor submitted by artificial intelligence. The main task is to determine a trajectory with continuous curvature to ensure continuous steering angles. For this purpose, a spline algorithm was implemented, which results in an optimized trajectory with a continuous curvature.

Caroline is designed to follow this track. Vehicle guidance is separated into two parts: A soft real-time and a hard real-time software module. The soft real-time module communicates with the artificial intelligence. Also, it converts the drivable corridor into segments. The hard real-time module contains the control algorithm that guides Caroline along the trajectory. It generates CAN messages that are executed by the vehicle at a frequency of 100 Hz.

For the lateral control part, two different control strategies were examined. Initially, an approach based on a tricycle model was evaluated. Without considering the dynamics of the car, the angle of the front wheels is calculated so that the front wheels point to a position on the track in front of the car. The distance of this position with respect to Caroline is a function of current vehicle speed. The control-algorithm described works well for lower speeds. However, neglecting vehicle dynamics, this approach has difficulties to keep the car on the desired track at higher

speeds particularly in twisting areas. As the final controller, a map-based pilot control algorithm in combination with different control loops is used (Figure 9). It keeps the vehicle on track with a minimum of deviation. The map-based pilot control algorithm calculates the steering angle that would be needed to follow the desired track based on parameters of the bicycle model.

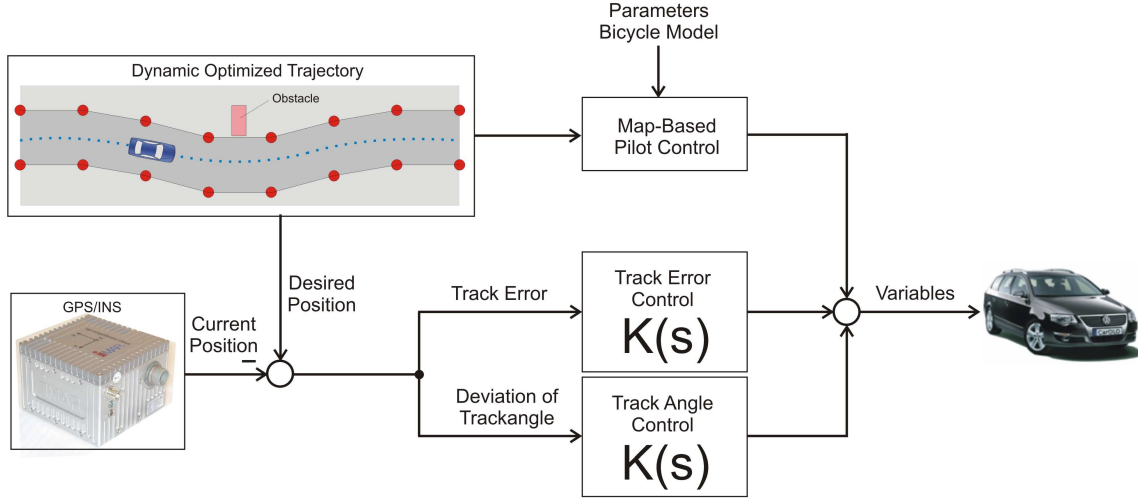


Figure 9: Final Control Strategy

As a result, the controllers for track error and track angle must control the remaining deviations to zero. For the longitudinal control of the car, a PI-controller is used in combination with a known engine map to determine and maintain a certain speed. A state-machine deals with the interaction of the brake and gas pedal. The control module allows forward as well as backward motion.

4.7 Supervisory Watchdog and Online-Diagnosis

The safety systems of Caroline must ensure the highest possible safety for the car and the environment in manned as well as unmanned operation. These systems must monitor the integrity of all appropriate hardware and software components. In case of an error, the systems must bring the car to a safe stop. Additionally they must provide an interface for pausing or disabling the car via remote control (E-Stop). We extended these basic features by providing the ability to reset and restart separate modules, independently using hardware and/or software means for automated failure correction. Figure 10 depicts the watchdog concept.

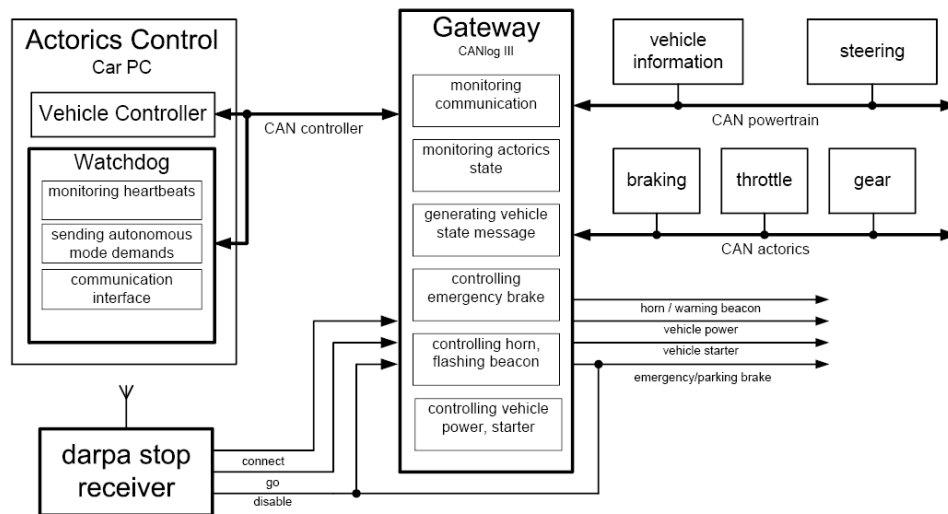


Figure 10 watchdog architecture

Caroline is outfitted with two separate brake systems, the standard hydraulic braking system of the Passat and an additional electronic parking brake. The main hydraulic brake is controlled by pressure, usually generated by the driver through the brake pedal. In autonomous mode, this pressure is generated by a small hydraulic brake booster. The parking brake is controlled by a push button in the instrument panel. This electronic brake offers an additional and useful feature. If the button is pressed while the car is rolling, the main brake system is activated in addition to the parking brake until the car comes to a complete stop. During autonomous mode, the watchdog gateway, the emergency buttons on the top of the car and the receiver for the DARPA E-stop form a safety circuit, which holds a safety relay open. This relay is connected to the push button for the electronic parking brake. If one of the systems should fail, or is activated, the safety circuit is opened, the relay contact is closed and the push button for the parking brake is activated. During emergency braking, the lateral controller of the car is still enabled to hold the car on the given course.

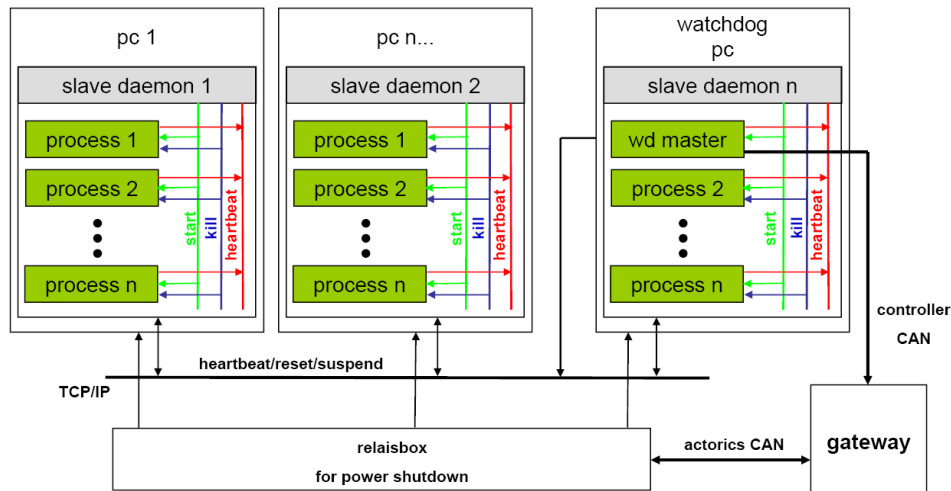


Figure 11 software watchdog concept

4.8 Telemetry and Data Storage for Offline Analysis

Telemetry data is processed by a visualization engine, enabling developers to manage complex data, such as planned trajectories and detected obstacles that can be displayed in an aerial image of the vehicle's current surroundings. This integration of static aerial view overlaid with dynamic objects results in a very intuitive user interface to the sensor data. An additional opportunity to store and replay the data exchanged between all software modules enables the developers to carry out offline-simulations based on real data.

5 Performance Evaluation

Because our vehicle only became fully operational in April 2007, detailed performance evaluations have not yet been finalized. At present we are focused on improving system robustness and developing test procedures for future performance evaluations, thus only preliminary results are available at this time.

5.1 Sensor Data Acquisition and Fusion

Performance of the data fusion unit can be directly measured by the object data throughput and the reaction time of the chain formed by sensor data acquisition and sensor data fusion. The sensor system operates at a medium cycle time of 80ms when loading the data fusion unit with incoming sensor sweeps consisting of several sensor objects. The quantity of incoming data is obviously situation-dependent considering the complexity of the vehicle's surroundings. According to the sequential data fusion scheme, each sensor is fused immediately after measurement reception. To cancel out clutter, the pre-tracking stage checks whether sensor readings stabilize over time. In the ideal case, a measurement seen by all front sensors (radar, lidar and laser scanners) becomes an active track after 5 update cycles which adds up to an overall time of approximately 160ms (two complete sensor cycles) from the first measurement up to track creation. This delay is used to cancel out false alarms and significantly increase track quality, so that a balance between false-detections and detection speed can be found. Another interesting performance criterion is processor load of both the data acquisition and fusion units. While a centralized fusion running on a single computation unit loaded by all sensors leads to a processor load of more than 80 percent at peak (complex situations), the splitting of the fusion system into front and back section balances this load equally onto the two processors and therefore leaves enough space for further extensions.

5.2 Image Processing

Because performance in terms of frames per second is a critical issue in image processing, two decisions were made concerning the image processing pipeline:

- All operations are applied to the entire image, i.e. the transformation and the merging of all three camera images is performed by a GPU to maintain a high image processing rate. Due to the large amount of data uploaded to and downloaded from the graphics card, we used a Nvidia GeForce 7600 card connected via an express PCI slot.

- All complex operations, such as feature detection, are restricted to the one-dimensional scanlines rather than to the entire image. Thus, the computation time for these operations becomes negligible.

The GPU enables us to process three camera images at the full frame-rate of 30 frames per second. In comparison, using the OpenCV library, a single camera can be processed on the CPU at approximately 15 frames per second on a 2 GHz Pentium machine.

5.3 Digital Map

We developed a benchmark scenario in order to be able to compare the performance of different map approaches. This simulation contained a variable number of fusion objects in Caroline's field of view. Three additional fusion objects are located in the middle of the driving corridor forcing Caroline to carry out evasion maneuvers. Point of interest has been the combined reaction time of map and AI.

The results of these measurements are presented in Figure 12. The fluctuation in the object-based approach is caused by the separation of fusion objects in- and outside of the sphere of interest. The grid produces a clearly higher basic load by erasing the layers even without obstacles, but shows a smaller upward gradient. There is no major time difference between the two approaches if there are many objects. For future speed issues, it is conceivable that a hybrid approach can be used together with an optimized grid handling for static obstacles and an object-based part for the dynamic obstacles.

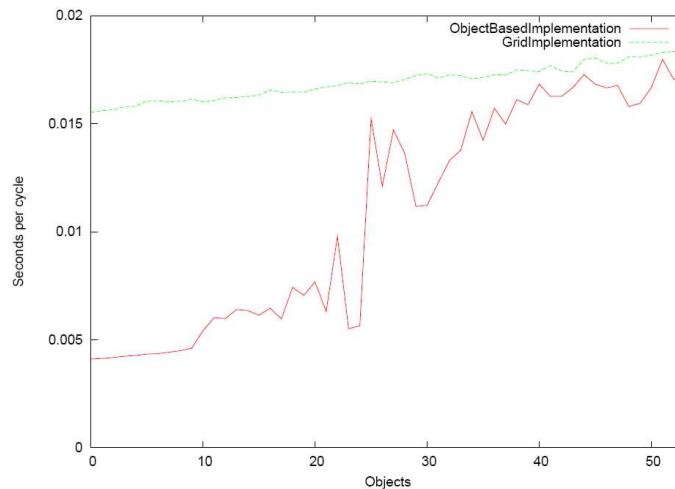


Figure 12 Comparison of different map implementations

5.4 Vehicle Path Planning and Low Level Control

With respect to lateral control, Figure 13 compares two different control strategies. Both, the controller, based on the tricycle model, as well as our final control strategy are evaluated on a given track at a vehicle speed of 10 m/s. To point out the performance of both controllers, Figure 13 shows the track error of the vehicle with respect to the desired trajectory.

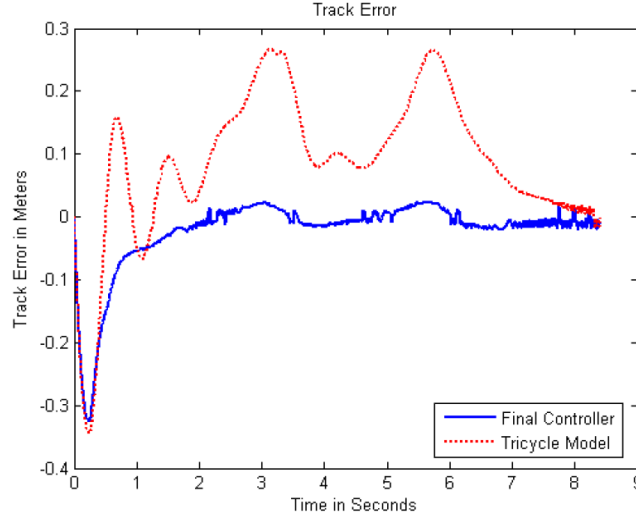


Figure 13 Comparison of the Track Error of Two Different Control Strategies

6 Testing

We implemented a multi-level test process using elements of extreme programming [2] consisting of a virtual test environment shown in Figure 14 and field tests described below.

The work flow for checking and releasing software formally consists of five consecutive steps and one optional step. First, the source is compiled to check for syntactical errors. While running the test code, the memory leak checker valgrind [8] checks for existing and potential memory leaks in the source code. After the execution of the test code, the coverage of the source code is computed by simply counting the executed statements. The intent is to implement test cases that completely cover the existing source code. The last step is for optimization purposes only and executes the code in order to find time-consuming parts inside an algorithm. The step shown on the right side of the schematic drawing depicts a compliance check for the coding guidelines. For example, it analyses the source for the naming of variables and classes. This process is also automatically executed after every commitment to our versioning system Subversion [4] by a central server.

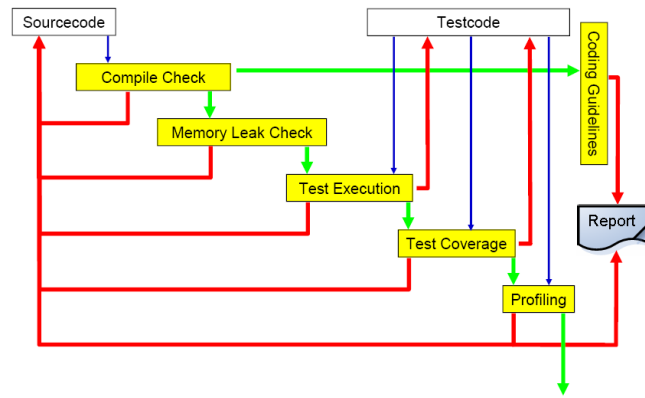


Figure 14: Workflow for testing and releasing software.

A novel approach for testing and assuring a high level of quality for several interacting components is accomplished on a second server running complex functional and non-functional test cases for several system components [7]. The functional test cases independently start the necessary software components and provide appropriate RND- and MD-files for specific scenarios. To test the AI, a simple simulator for emulating the missing parts of the complete software chain is necessary. This architecture can be extended with plug-ins for validating various aspects, such as waypoint following (Figure 15). One central generator computes the current vehicle state including position, speed and rotation taking into account the planned trajectory, velocity and elapsed time. This information is broadcast to every listener connected to the simulator. One listener is the AI that analyzes the vehicle's state to produce new trajectories. Additional validators receive all states and perform compliance checks with respect to previously defined reference behavior. Additional modules for simulating detected lanes, static/dynamic obstacles are also available. To check the behavior of the AI concerning the given RND file and the actual lane, a second RND file can be passed over to the simulator. This additional and independent RND file is used to provide lane data, which is normally detected by the image processing algorithm. In order to be able to check relevant software modules for their interaction with dynamic obstacles, an advanced concept has been developed. This approach is similar to the one used for providing detected lanes. Dynamic Obstacles interact based on their own individual RND- and MD-files.

To allow for the quick and convenient creation of test scenarios, various concepts and tools have been developed. A GUI tool for creating RND files further simplifies the creation of test scenarios.

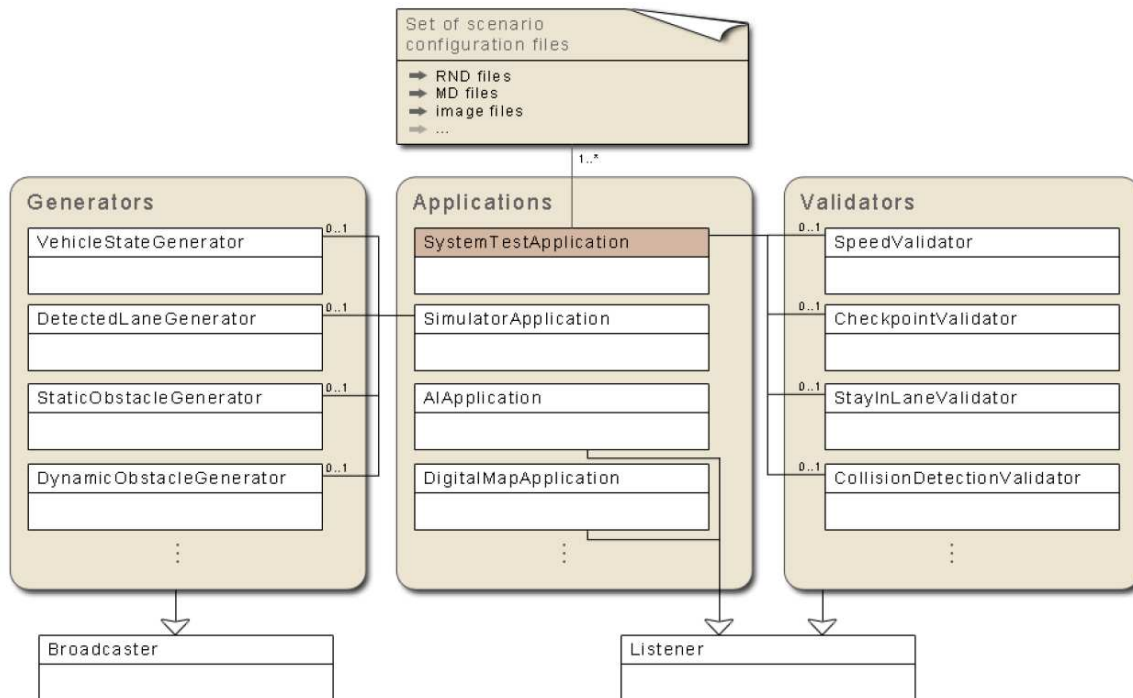


Figure 15 System test and simulator architecture.

Real world tests are derived from the requirements of the DARPA Rules [5]. The results are transferred into a web-based bug tracking system for documenting, assigning and fixing bugs. Therefore, a specially developed recorder captures every data object and writes it to a file. This file is attached to a bug report and can be used for replaying a specific situation during a real test drive. The remaining bugs are scheduled into our weekly story card for milestone and project tracking and planning.

7 Summary

Team CarOLO is an interdisciplinary team made up of members from several university departments and industrial sponsors. Our vehicle Caroline is a standard 2006 Volkswagen Passat built to European specifications that is able to detect and track obstacles at a distance of up to 200m. The system's architecture comprises eight main modules: sensor data acquisition, sensor data fusion, image processing, digital map, artificial intelligence, path planning, watchdog and diagnosis, and telemetry. The signal flow through these modules is generally linear in order to decouple the development process.

Our design approach uses multi-sensor fusion of lidar, radar and laser scanners, extending the classical point shape based approach to handle extensive dynamic targets expected in urban environments. Image processing detects road markings and drivable areas. Fusion objects and road markings are stored in and processed by a digital map module. Artificial intelligence is modeled according to DAMN architecture, redesigned and enhanced to meet requirements of urban environments. Our approach is able to handle complex situations and ensure Caroline's proper behavior, e.g. obeying traffic regulations at intersections or performing u-turns when roads are blocked. AI decisions are communicated to the path planner, which calculates optimal vehicle trajectories with respect to its dynamics. Safety and robustness is ensured by supervisory watchdog monitoring of all vehicle hardware and software modules. Failures or malfunctions immediately result in a safe and complete stop by Caroline.

Since we are a large heterogeneous team with a very tight project schedule, we recognized very early the need for efficient quality assurance during the development process. Thus, we implemented an automatic multi-level test process. Each new feature or modification runs through a series of unit tests before being deployed on the vehicle. Caroline is able to autonomously perform missions containing all elements required by the Site Visit Guidelines.

References

- [1] Yaakov Bar-Shalom, „Estimation and Tracking”, Artech House, 1993.
- [2] K. Beck, “Extreme Programming Explained: Embrace Change”. Addison Wesley, 2005.
- [3] J.-C. Becker, „Fusion der Daten der objekterkennenden Sensoren eines autonomen Straßenfahrzeugs”, Ph.D. thesis, Institute of Control Engineering, Braunschweig, Germany, March 2002.
- [4] B. Collins-Sussmann, B. W. Fitzpatrick, and C. M. Pilato, “Version Control with Subversion”. O’Reilly, 2004.
- [5] DARPA, “Technical evaluation criteria,” 3 2006. [Online]. Available: http://www.darpa.mil/grandchallenge/docs/Technical_Evaluation_Criteria_031607.pdf
- [6] W. E. Lewis and G. Veerapillai, “Software Testing and Continuous Quality Improvement”, Second Edition. Auerbach Publications, 2004.
- [7] P. Liggesmeyer, Software-Qualität : “Testen, Analysieren und Verifizieren von Software”. Spektrum, Akad. Verl., 2002.
- [8] N. Nethercote and J. Seward, “Valgrind: A program supervising framework,” Theoretical Computer Science, vol. 89, 2003.
- [9] M. Pellkofer, “Verhaltensentscheidung für autonome Fahrzeuge mit Blickrichtungssteuerung (behavior decision for autonomous vehicles with gaze control),” Ph.D. dissertation, Institut für Systemdynamik und Flugmechanik, Universität der Bundeswehr München, January 2003.
- [10] J. Rosenblatt, “Damn: A distributed architecture for mobile navigation,” Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1997.